Application: Set Membership

We have a set $S \subset \mathcal{D}$, $s = |S| << |\mathcal{D}|$. We need a small data structure and fast algorithm for testing "$y \in S?$" queries.

We use hash function $h : \mathcal{D} \to \{0, \ldots, m-1\}$:

- for each $x \in \mathcal{D}$: $\Pr(h(x) = j) = \frac{1}{m}$;
- values of $f(x)$ for each x are independent.

We store the collection of fingerprints $F(S, h) = \{h(x) \mid x \in S\}$ in a sorted order. To check if $y \in S$ we run binary search for $h(y)$ in $F(S, h)$.

- space: $s \log m$ bits ($\log m$ bits per element of S).
- time: $O(\log s)$.

Application: Set Membership

We store the collection of fingerprints $F(S, h) = \{h(x) \mid x \in S\}$ in a sorted order. To check if $y \in S$ we run binary search for $h(y)$ in $F(S, h)$.

- space: $b = s \log m$ bits ($\log m$ bits per element of S).
- time: $O(\log s)$.

Problem False positives: $y \notin S$, but $h(y) = h(x)$ for some $x \in S$.

Probability of false positive $\leq \frac{s}{m} = \frac{s}{2^{b/s}}$. Therefore we must use $b \geq s \log s$ bits.
If we use $b = 2s \log s$ bits (that is, $2 \log s$ bits per element of S) then probability of false positive $\leq \frac{s}{2^{2s \log s/s}} = \frac{1}{s}$.

Bloom Filter

Choose k hash functions $h_i : \mathcal{D} \to \{0, \ldots, b-1\}$.

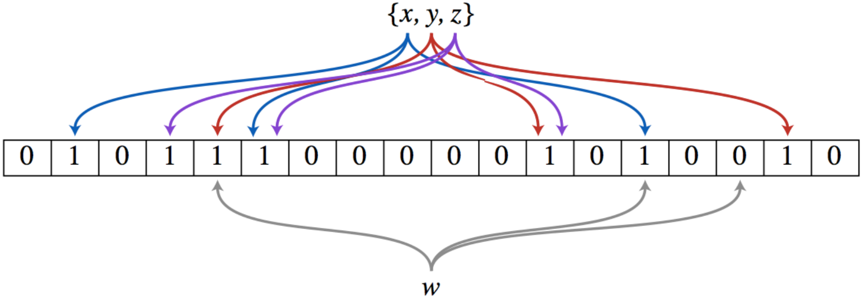
Let $A[0, b-1]$ be a binary array of b entries.

For each $x \in S$ set the k bits $A[h_1(x)], \ldots, A[h_k(x)]$ to 1.

To test $y \in S$, output YES if and only if all bits $A[h_1(y)], \ldots, A[h_k(y)]$ are set to 1.

Bloom Filter

S = {x,y,z}; k = 3



Bloom Filter: False Positive

The probability that $A[i] = 0$ is $\left(1 - \frac{1}{b}\right)^{sk} \approx e^{-sk/b}$

Set $k = \frac{b}{s} \ln \frac{3}{2}$ then $Pr(A[i] = 1) \approx \frac{1}{3}$.

Let X be the number of bits set to 1. With high probability $X \leq \frac{b}{2}$.

Probability of false positive $\leq \left(\frac{1}{2}\right)^{k} \leq \left(\frac{1}{2}\right)^{\frac{b}{s} \ln \frac{3}{2}}$. Therefore we need $b \geq s$ bits.
Taking $b = cs$ for a small constant (say 20) we can obtain a false positive probabiklity of less than 0.01.
Thus Bloom filters allow us to use much fewer bits per word than fingerprinting (of course at the price of using several (but a constant number!! of) hash functions)

Symmetry breaking

Typical problem in distributed computing: n users want to use a resource and only one can use it at any given time.
How can we decide a permutation of the users quickly and fairly?
Idea: hash each users id into 2^b bits and then take the permutation given by the sorted order of the resulting numbers
Problem: we need to avoid that two user ids hash to the same value.

Symmetry breaking

Look at the situation from a fixed user i :
The probability that one of the $n-1$ other users obtain the same hash value as i is

$$1-(1-\frac{1}{2^b})^{n-1} \leq \frac{n-1}{2^b} \tag{1}$$

So by the union bound, the probability that any two users get the same hash value is at most $\frac{n(n-1)}{2^b}$.
Hence choosing $b = 3\log_2 n$ guarantees succes with probability at least $1-\frac{1}{n}$.

Random Graphs

Many important computation problems are defined on graphs.

Many of these problems are NP-Complete but are solved "efficiently" in practice.

A probabilistic model of graphs for probabilistic analysis of graph algorithms.

Random Graph Process

Consider the following stochastic process:

- Start with n vertices, no edges.
- In each step add one edge between a randomly chosen pair of vertices.

If we stop the process after N steps (i.e. after adding N random edges):

1. Is the graph connected?
2. Does the graph have a large connected component?
3. Are there isolated vertices?

A graph property is **monotone** if for any two graphs $G = (V, E)$ and $G' = (V, E')$, such that $E \subseteq E'$, if G has the property also G' has that property.

Monotone properties:

1. No isolated vertices;
2. Connectivity;
3. Perfect Matching;
4. Hamiltonian Path;
5. ....

The $G_{n,N}$ model:

- The set of all graphs on n vertices with exactly N edges.
- All graphs in this set have equal probability.
- There are $T = \binom{\binom{n}{2}}{N}$ graphs on n vertices with exactly N edges.
- The probabilistic space $G_{n,N}$ has T simple events, each with probability $\frac{1}{T}$.

1. What is the probability that a graph in $G_{n,N}$ has isolated vertices?
2. What is the probability that a graph in $G_{n,N}$ is connected?
3. What is the probability that a graph in $G_{n,N}$ has a Hamiltonian cycle?
4. How fast can an algorithm find a Hamiltonian cycle in $G \in G_{n,N}$?

Isolated Vertices

Theorem

Let $N = \frac{1}{2}(n \log n + cn)$*, and let* $P_v(n, N)$ *be the probability that* $G \in G_{n,N}$ *has isolated vertices, then* $\lim_{n\to\infty} P_v(n, N) = 1 - e^{-e^{-c}}$.

Proof.

View the two endpoints of an edge as two balls placed uniformly at random into n boxes (nodes of the graph).
The number of isolated nodes is the number of empty boxes.

$$E[\text{number of empty boxes}] = n\left(1 - \frac{1}{n}\right)^{2N} \leq ne^{-(\log n + c)} = e^{-c}.$$

The number of empty boxes is distributed Poisson with $\lambda = e^{-c}$.
Probability of 0 empty boxes $= e^{-\lambda} = e^{-e^{-c}}$

□

Isolated Vertices II

Theorem

Let $N = \frac{1}{2}(n \log n + cn)$, *and let* $P_v(n, N)$ *be the probability that* $G \in G_{n,N}$ *has isolated vertices, then* $P_v(n, N) \leq e^{-c}$.

Proof.

"Coupon collector" argument:

$$P_v(n, N) \leq n \left(1 - \frac{1}{n}\right)^{2N} \leq e^{-c}$$

. $\square$

Connectivity

Theorem

Let $N = \frac{1}{2} n \log n + w(n) n$*, and let* $P_c(n, N)$ *the probability that* $G \in G_{n,N}$ *is connected. As* $n \to \infty$*:*

$$P_c(n, N) \to \begin{cases} 0 & \text{if } w(n) \to -\infty \\ 1 & \text{if } w(n) \to \infty \end{cases}$$

The $G_{n,p}$ model:

- The set of all graphs on n vertices.
- The probability of a graph with M edges is $p^M(1-p)^{\binom{n}{2}-M}$
- Let $G \in G_{n,p}$. Given that G has M edges it has the same distribution as $G \in G_{n,M}$.
- For $N = \binom{n}{2}p$
  - $G_{n,N}$ and $G_{n,p}$ have similar monotone properties.
  - Let $G_{n,p}$ with $p \geq 1/n$, and M edges. There is $c > 0$, such that

$$Pr(M \in [N - c\sqrt{N}, N + c\sqrt{N}] \geq 1 - 1/n$$

    If a property holds with probability $\leq R$ in $G_{n,p}$ it hold with probability $\leq c\sqrt{M}R$ in $G_{n,M}$.

Isolated Vertices III

Theorem

Let $p = \frac{\log n + c}{n}$*, and let* $P_v(n,p)$ *be the probability that* $G \in G_{n,p}$ *has isolated vertices, then* $\lim_{n\to\infty} P_v(n,p) = 1 - e^{-e^{-c}}$.

Theorem

Let $p = \frac{\log n + c}{n}$*, and let* $P_v(n,p)$ *be the probability that* $G \in G_{n,p}$ *has isolated vertices, then* $P_v(n,p) \leq e^{-c}$.

Algorithm for Finding a Hamiltonian Path

A **simple** path is a path with no loops, i.e. a vertex is visited no more than once.

A **Hamiltonian Path** is a simple path that visits every vertex of the graph.

A **Hamiltonian Cycle** is a cycle that visits every vertex in the graph exactly once.

Given a graph G, deciding if G has a Hamiltonian path/cycle is NP-Complete.

Theorem

Let G be a graph chosen randomly from $G_{n,p}$ for $p \geq \frac{c \log n}{n}$ with some constant $c > 0$. There is an $O(n \log n)$ algorithm that finds, with high probability, a Hamiltonian path (cycle) in G.

Rotation

Let G be an undirected graph. Assume that

$$P = v_1, v_2, ..., v_k$$

is a simple path in G and (v_k, v_i) is an edge of G then

$$P' = v_1, ..., v_i, v_k, v_{k-1}, ..., v_{i+2}, v_{i+1}$$

is a simple path is G.

Algorithm

Assume that each vertex has its list of adjacent edges, in a random order.

1. Choose an arbitrary vertex x_0 to start the path. $HEAD = x_0$.
2. Repeat until all vertices are connected
    1. Let $(HEAD, u)$ be the first edge in $HEAD$'s list.
    2. Remove $(HEAD, u)$ from $HEAD$'s and u's lists.
    3. If u not in the path $HEAD := u$, else use the edge to "rotate" the path.

"Almost" a coupon collector paradigm.
Can we modify the algorithm so that at each step the HEAD is chosen uniformly and independently from all the nodes?

Modified Algorithms

Consider a "less efficient" algorithm that for each vertex u keeps two lists:

1. $unused\_edges(u)$ - adjacent edges that were not used yet;
2. $used\_edges(u)$ - edges that were already used.

When u is at the head of the path we choose

- a random element in $used\_edges(u)$, with probability $\frac{|used\_edges(u)|}{n}$;
- the tail of the path becomes the head of the path, with probability $\frac{1}{n}$;
- the head of $unused\_edges(u)$ list (and move it to $used\_edges(u)$), otherwise (with probability $1 - \frac{1}{n} - \frac{|used\_edges(u)|}{n}$).

Lemma

The probability that a given vertex becomes HEAD at a given iteration of the modified algorithm is $\frac{1}{n}$.

Proof.

Clear for the tail of the paths and for neighbors of the current head in old edges.
The probability of using the $unused\_edge(u)$ list is

$$1 - \frac{1}{n} - \frac{|used\_edges(u)|}{n}$$

and that edge is connected to a vertex chosen uniformly at random from a set of

$$n - 1 - |used\_edge(u)|$$

vertices. □

Are choices in successive steps independent?

Independent unused-edge lists

Let $q \in [0, 1]$ such that $p = 2q - q^2$.

(Initialization) For any edge (u, v) do exactly one of the following:

1. With probability $q(1-q)/(2q-q^2)$, place the edge on u's unused edge-list, but not v's;
2. With probability $q(1-q)/(2q-q^2)$, place the edge on v's unused edge-list, but not u's;
3. With probability $q^2/(2q-q^2)$, the edge is placed on both unused-edge lists.

For edge (x, y), the probability that it is initially placed in the unused-edge list for x is

$$p\left(\frac{q(1-q)}{2q-q^2} + \frac{q^2}{2q-q^2}\right) = q;$$

The probability that it is placed in both x's and y's lists is:

$$\frac{pq^2}{2q-q^2} = q^2,$$

so events are independent.

Modified Hamiltonian Cycle Algorithm:

1. Start with a random vertex as the head of the path.
2. Repeat until the rotation edge closes a Hamiltonian cycle or the unused-edges list of the head of the path is empty:
   1. Let the current path be $P = v_1, v_2, \ldots, v_k$, with v_k being the head.
   2. Execute i, ii or iii below with probabilities $\frac{1}{n}$, $\frac{|\text{used-edges}(v_k)|}{n}$, and $1 - \frac{1}{n} - \frac{|\text{used-edges}(v_k)|}{n}$, respectively:
      1. Reverse the path, and make v_1 the head.
      2. Choose uniformly at random an edge from $\text{used-edges}(v_k)$; if the edge is (v_k, v_i), rotate the current path with (v_k, v_i) and set v_{i+1} to be the head. (If the edge is (v_k, v_{k-1}), then no change is made.)
      3. Select the first edge from $\text{unused-edges}(v_k)$, call it (v_k, u). If $u \neq v_i$ for $1 \leq i \leq k$, add $u = v_{k+1}$ to the end of the path and make it the head. Otherwise, if $u = v_i$, rotate the current path with (v_k, v_i), and set v_{i+1} to be the head. (This step closes the Hamiltonian path if $k = n$ and the chosen edge is (v_n, v_1).)
   3. Update the used-edges and unused-edges lists appropriately.
3. Return a Hamiltonian cycle if one was found or failure if no cycle was found.

Theorem

Suppose the input to the modified Hamiltonian cycle algorithm initially has unused edge-lists where each edge (v, u) with $u \neq v$ is placed on v's list independently with probability $q \geq \frac{20 \ln n}{n}$. Then the algorithm successfully finds a Hamiltonian cycle in $O(n \ln n)$ iterations of the repeat loop (step 2) with probability $1 - O(n^{-1})$.

Note that we did not assume that the input random graph has a Hamiltonian cycle.

$\mathcal{E}_1$: The algorithm run $3n \ln n$ iterations with no unused-edges list becoming empty, but failed to construct a Hamiltonian cycle.
$\mathcal{E}_2$: At least one unused-edges list became empty during the first $3n \ln n$ iterations of the loop.
We first bound $\Pr(\mathcal{E}_1)$.
The probability that any vertex was not chosen in $2n \ln n$ iterations is at most

$$n\left(1 - \frac{1}{n}\right)^{2n \ln n} \leq n\mathrm{e}^{-2\ln n} = \frac{1}{n}.$$

The probability that the path does not become a cycle within the next $n \ln n$ iterations is

$$\left(1 - \frac{1}{n}\right)^{n \ln n} \leq \mathrm{e}^{-\ln n} = \frac{1}{n}.$$

$$\Pr(\mathcal{E}_1) \leq \frac{2}{n}.$$

$\Pr(\mathcal{E}_2) =$ the probability that an unused-edges list is empty in the first $3n \ln n$ iterations.
$\mathcal{E}_{2a}$: At least $9 \ln n$ edges were removed from the unused-edges list of at least one vertex in the first $3n \ln n$ iterations of the loop.
$\mathcal{E}_{2b}$: At least one vertex has fewer than $10 \ln n$ edges.

$$\Pr(\mathcal{E}_2) \leq \Pr(\mathcal{E}_{2a}) + \Pr(\mathcal{E}_{2b}).$$

We bound $\Pr(\mathcal{E}_{2a})$.

Let X_j^i be a Bernoulli random variable that is 1 if the i-th vertex is adjacent to the edge used in the j-th iteration of the loop and 0 otherwise.

$X^i = \sum_{j=1}^{3n \ln n} X_j^i$.

$\mathbf{E}[X_j^i] = \frac{1}{n}$ and $\mathbf{E}[X^i] \leq 3 \ln n$.

$$\Pr(X^i \geq 9 \ln n) \leq \left(\frac{e^2}{27}\right)^{3 \ln n} \leq \frac{1}{n^2}.$$

$\Pr(\mathcal{E}_{2a}) \leq 1/n$.

$\mathcal{E}_{2b}$: At least one vertex has $10 \ln n$ or fewer edges initially in its unused-edges list.
$Y^i =$ number of edges initially in vertex i unused-edges list.
$\mathbf{E}[Y^i] = (n-1)q \geq 20(n-1)\ln n/n \geq 19 \ln n$ for sufficiently large n.

$$\Pr(Y^i \leq 10 \ln n) \leq \mathrm{e}^{-19 \ln n (9/19)^2/2} < \frac{1}{n^2}$$

$$\Pr(\mathcal{E}_{2b}) < \frac{1}{n},$$

$$\Pr(\mathcal{E}_2) \leq \frac{1}{n} + \frac{1}{n} = \frac{2}{n}.$$

$$\Pr(\mathcal{E}_1) + \Pr(\mathcal{E}_2) \leq \frac{2}{n} + \frac{2}{n} = \frac{4}{n}.$$

Corollary

By putting edges on the unused-edges lists appropriately, the algorithm finds a Hamiltonian cycle on a graph chosen randomly from $G_{n,p}$ with probability $1 - O(1/n)$ whenever $p \geq 40 \ln n/n$.

We need $q \in [0, 1]$ such that $p = 2q - q^2$, and $q \geq 20 \ln n/n$.
If $p \geq 40 \ln n/n$ then $q \geq p/2 \geq 20 \ln n/n$.